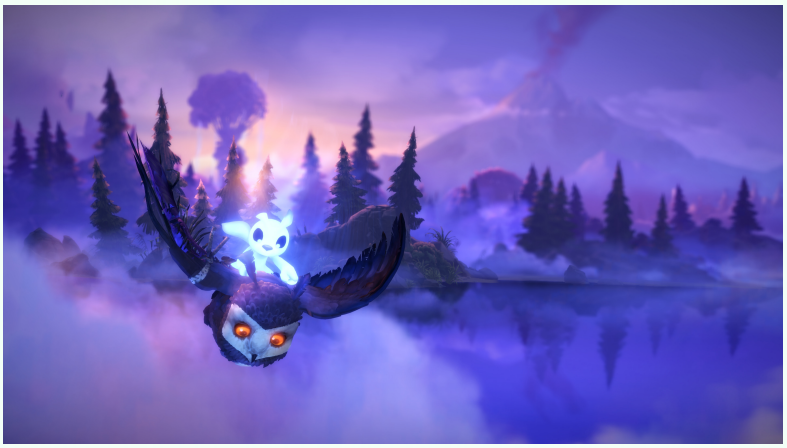
Computer Organization and Architecture

Note

计算机组成原理 笔记



计算机是计算机科学的心脏————没有计算机，计算机科学只能作为理论数学的一个分支。

整理：XYL & XYL

整理时间：March 23, 2025

Email: xyl_27@outlook.com

Version: 1.0.0

目　　录

第 1 章
大纲

1.1 计算机系统概述

1.1.1 计算机系统层次结构

1. 计算机系统的基本组成
2. 计算机硬件的基本结构
3. 计算机软件和硬件的关系
4. 计算机系统的工作原理

”存储程序”工作方式，高级语言程序与机器语言程序之间的转换，程序和指令的执行过程

1.1.2 计算机性能指标【21 增】

吞吐量、响应时间；CPU 时钟周期、主频、CPI、CPU 执行时间；MIPS、MFLOPS、GFLOPS、TFLOPS、PFLOPS、EFLOPS、ZFLOPS

1.2 数据的表示和运算

1.2.1 数制与编码

1. 进位计数制及其数据之间的相互转换
2. 定点数的编码表示

1.2.2 运算方法和运算电路

1. 基本运算部件：加法器、算术逻辑都件 (ALU)
2. 加减运算：补码加减运算器，标志位的生成
3. 乘除运算：乘除运算的基本原理，乘法除法电路的基本结构

1.2.3 整数的表示和运算

1. 无符号整数的表示与运算

2. 带符号整数的表示与运算

1.2.4 浮点数的表示和运算

1.3 存储器层次结构

1.3.1 存储器的分类

1.3.2 层次化存储器的的基本结构

1.3.3 半导体随机存取存储器

1.SRAM 存储器
2.DRAM 存储器
3.Flash 存储器

1.3.4 主存储器

1.DRAM 芯片和内存条
2. 多模块存储器
3. 主存和 CPU 之间的连接

1.3.5 外部存储器【22 增】

1. 磁盘存储器
2. 固态硬盘 (SSD)

1.3.6 高速缓冲存储器 (Cache)

1.Cache 的基本工作原理
2.Cache 和主存之间的映射方式
3.Cache 中主存块的替换算法
4.Cache 写策略

1.3.7 虚拟存储器

1. 虚拟存储器的基本概念
2. 页式虚拟存储器
基本原理，页表，地址转换，TLB (快表)
3. 段式虚拟存储器的基本原理
4. 段页式虚拟存储器的基本原理

1.4 指令系统【22 增】

1.4.1 指令系统的基本概念

1.4.2 指令格式

1.4.3 寻址方式

1.4.4 数据的对齐和大/小端存放方式

1.4.5 CISC 和 RISC 的基本概念

1.4.6 高级语言程字与机器级代码之间的对应

1. 编译器、汇编器和链接器的基本概念
2. 选择结构语句的机器级表示
3. 循环结构语句的机器级表示
4. 过程 (函数) 调用对应的机器级表示

1.5 中央处理器 (CPU)

1.5.1 CPU 的功能和基本结构

1.5.2 指令执行过程

1.5.3 数据通路的功能和基本结构

1.5.4 控制器的功能和工作原理

1.5.5 异常和中断机制

1. 异常和中断的基本概念
2. 异常和中断的分类
3. 异常和中断的检测与响应

1.5.6 指令流水线

1. 指令流水线的基本概念
2. 指令流水线的基本实现

3. 结构冒险、数据冒险和控制冒险的处理
4. 超标量和动态流水线的基本概念

1.5.7 多处理器基本概念【22 增】

1.SISD、SIMD、MIMD、向量处理器的基本概念
2. 硬件多线程的基本概念
3. 多核处理器 (multi-core) 的基本概念
4. 共享内存多处理器 (SMP) 的基本概念

1.6 总线和输出输出系统

1.6.1 总线概述

1. 总线的基本概念
2. 总线的组成及性能指标
3. 总线事务和定时

1.6.2 I/O 接口 (I/O 控制器)

1.I/O 接口的功能和基本结构
2.I/O 端口及其编址

1.6.3 I/O 方式

1. 程序查询方式
2. 程序中断方式
中断的基本概念，中断响应过程，中断处理过程，多重中断和中断屏蔽的概念
3.DMA 方式
DMA 控制器的组成，DMA 传送过程

第 2 章
计算机系统概述

RTL 指令

2.1 计算机系统层次结构

- 08.MAR 和 MDR 的位数分别为（）
  A. 地址码长度、存储字长
  B. 存储字长、存储字长
  C. 地址码长度、地址码长度
  D. 存储字长、地址码长度

  地址寄存器（MAR）存放访存地址，因此位数与地址码长度相同。数据寄存器（MDR）用于暂存要从存储器中读或写的信息，因此位数与存储字长相同。

- 20. 关于相联存储器，下列说法中正确的是（） A. 只可以按地址寻址
  B. 只可以按内容寻址
  C. 既可按地址寻址又可按内容寻址
  D. 以上说法均不完善

  相联存储器既可以按地址寻址又可以按内容（通常是某些字段）寻址，为与传统存储器区别也称按内容寻址的存储器

- 21.【2015 统考真题】计算机硬件能够直接执行的是（）
  I. 机器语言程序 II. 汇编语言程序 III. 硬件描述语言程序
  A. 仅 I
  B. 仅 I、II
  C. 仅 I、III
  D.I、II、III
  A

- 23.【2019 统考真题】下列关于冯·诺依受计算机基本思想的叙述中，错误的是（）。
  A. 程序的功能都通过中央处理器执行指令实现

B. 指令和数据都用二进制数表示，形式上无差别
C. 指令按地址访问，数据都在指令中直接给出
D. 程序执行前，指令和数据需预先存放在存储器中

冯·诺依曼结构计算机的功能部件包括输入设备、输出设备、存储器、运算器和控制器，程序的功能都通过中央处理器（运算器和控制器）执行指令，选项 A 正确。指令和数据以同等地位存放于存储器内，形式上无差别，只在程序执行时具有不同的含义，选项 B 正确。指令按地址访问，数据由指令的地址码指出，除立即寻址外，数据均存放在存储器内，选项 C 错误。在程序执行前，指令和数据需预先存放在存储器中，中央处理器可以从存储器存取代码，选项 D 正确。

- 24.【2022 统考真题】将高级语言源程序转换为可执行目标文件的主要过程是（）
A. 预处理 → 编译 → 汇编 → 链接
B. 预处理 → 汇编 → 编译 → 链接
C. 预处理 → 编译 → 链接 → 汇编
D. 预处理 → 汇编 → 链接 → 编译

A

2.2 计算机性能指标

- 08. 在计算机 M1 和计算机 M2 上分别运行功能完全相同的高级语言程序，程序在 M1 和 M2 上的平均 CPI 相等，则对于该类程序而言（）
A.MI 和 M2 执行速度相等
B.MI 和 M2 中主频高的计算机执行速度快
C.M1 和 M2 中主频低的计算机执行速度快
D. 无法确定哪会机器的执行速度快

CPU 执行时间 = 指令条数 ×CPIx 时钟周期，程序在 M1 和 M2 上的平均 CPI 相等，但影响 CPU 执行时间的因素还有指令条数和时钟周期，此外相同的高级语言程序在不同计算机上编译生成的机器指令条数可能不同，因此无法确定哪台机器执行该类程序的速度快。

- 13. 从用户观点看，评价计算机系统性能的综合参数是（）
A. 指令系统
B. 吞吐率
C. 主存容量
D. 主频率

主频、主存储器容量和指令系统（间接影响 CPI）并不是综合性能的体现。吞吐率指系统在单位时间内处理请求的数量，是评价计算机系统性能的综合参数。

- 14. 当前设计高性能计算机的重要技术途径是 ()
A. 提高 CPU 主频

B. 扩大主存容量

C. 采用非冯·诺依曼体系结构

D. 采用开行处理技术

提高 CPU 主频、扩大主存储器容量对性能的提升是有限度的。采用并行技术是实现高性能计算的重要途径，现今超级计算机均采用多处理器来增强并行处理能力。

- 【2012 统考真题】假定基准程序 A 在某计算机上的运行时间为 100s，其中 90s 为 CPU 时间，其余为 I/O 时间。若 CPU 速度提高 50%，IO 速度不变，则运行基准程序 A 所耗费的时间是（）。

A. 55s

B.60s

C.65S

D.70s

程序 A 的运行时间为 100s，减去 CPU 时间 90s，剩余 10s 为 I/O 时间。CPU 提速 50%后运行基准程序 A 所耗费的时间是 T=90÷1.5+10=70s。

第 3 章
计算机算术

3.1 二进制表示

3.1.1 二进制补码

二进制补码，$2^n - N$ 为位为 n 的 N 的补码. 注意补码的溢出。

$0\ 00000101 = +5$　　即$0 \times -128 + 0 \times 64 + 0 \times 32 + 0 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$

$= 4 + 1$

$1\ 11111011 = -5$　　即$1 \times -128 + 1 \times 64 + 1 \times 32 + 1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$

$= 64 + 32 + 16 + 8 + 2 + 1 - 128$

$= 123 - 128$

$= -5$

1. 负数右移需要特别注意。简单地将二进制补码负数 1110 0010 右移一位，结果为 0111 0001，这显然是不正确的。为了在移位时保持二进制补码数的符号不变，右移时应该复制符号位。考虑 1110 0010(即-30)。右移一位同时保持符号位不变将得到 1111 0001，等价于-15。

   为什么通过右移一位实现二进制补码数除以 2 时要在最高位补符号位? 二进制正数定义为 0xxxx,…,xx，这里 x 为 1 或 0。将该数除以 2，会得到 00pppp,…,pp。这个数的补数为 1yyyy…yy+1(这里每个 y 是对应的 x 的补)。现在把 00pppp,…,pp 转换为负数，会得到 11qqqq,…qq+1。正如我们所看到的那样，无论是正数还是负数，右移时符号位都应保持不变。

3.2 二进制的乘法

步骤 a. 将计数器的值置为 n。

步骤 b. 将 $2n$ 位的部分积寄存器清零。

步骤 c. 检查乘数的最右位（即最低位）。表 2-3 中用下划线标出了这一位，将被乘数与部分积的最低 n 位相加。

步骤 d. 将部分积右移一位。

步骤 e. 将乘数右移一位（乘数的最右位当然会被丢弃）。

步骤 f. 将计数器的值减 1，重复步骤 **c** 直到经过 n 个周期后计数器的值变为 0。部分积寄存器的内容就是乘积。

表 2-3　用图 2-3 中的方法计算无符号数乘法

周期	乘数 = 1101_2		被乘数 = 1010_2	
	步骤	计数值	乘数	部分积
	a 和 b	4	1101	00000000
1	c	4	1101	10100000
1	d 和 e	4	0110	01010000
1	f	3	0110	01010000
2	c	3	0110	01010000
2	d 和 e	3	0011	00101000
2	f	2	0011	00101000
3	c	2	0011	11001000
3	d 和 e	2	0001	01100100
3	f	1	0001	01100100
4	c	1	0001	10000010
4	d 和 e	1	0000	10000010

Note: 布斯乘法

3.3 二进制的除法

3.3.1 恢复余数

3.3.2 不恢复余数

3.4 浮点数

3.4.1 IEEE 浮点数

- 11. 若定点整数为 64 位，含 1 位符号位，则采用补码表示的绝对值最大的负数为 （） A.-2^{64}

B.$-2^{64}-1$

C.-2^{63}

D.$-2^{63}-1$

- 13. 若 $[x]_{补}=1,x_1x_2x_3x_4x_5x_6$, 其中 x_i 取 0 或 l, 若要 $x>-32$, 应当满足 ()。

A.x_1 为 0, 其他各位任意

B.x_1 为 l, 其他各位任意

C. x_1 为 l, $x_2\cdots x_6$ 中至少有一位为 1

D. x_1 为 $0,x_2\cdots x_6$ 中至少有一位为 1

- 14. 设 x 为整数，$[x]_{补}=1,x_1x_2x_3x_4x_5$, 若要$x<-16,x_1\sim x_5$ 应满足的条件是 ()。

A.$x_1\sim x_5$ 至少有一个为 1

B. x_1 必须为 $0,x_2\sim x_5$ 至少有一个为 l

C. x_1 必须为 $0,x_2\sim x_5$ 任意

D.x_1 必须为 1，$x_2\sim x_5$ 任意

- 15. 设 x 为真值，x* 为其绝对值，满足 [-x*] 补 =[-x] 补，当且仅当 ()。

A.x 任意

B.x 为正数

C.x 为负数

D. 以上说法都不对

- 23. 下列为 8 位移码机器数 [x] 移，求 [-x] 移时，() 将会发生溢出

A.11111111

B.00000000

C.10000000

D.01111111

- 24. 一个 8 位的二进制整数由 2 个“0”和 6 个“1”组成，采用补码或者移码表示，则说法中正确的是 ()。

A. 若采用移码表示，偏置值为 127，则此整数最小为-64

B. 若采用移码表示，偏置值为 128，则此整数最大为 123

C. 若采用补码表示，则此整数最小为-96

D. 若采用补码表示，则此整数最大为 252

- 27.16 位补码整数 0x8FAO 扩展为 32 位应该是（）

  A.0x0000 8FA0

  B.0xFFFF 8FA0

  C.0xFFFF FFA0

  D.0x8000 8FA0

第 4 章
图灵完备

4.1 优雅存储

锁存器（latch）

4.2 逻辑引擎

巧用德摩根定律

4.3 汉诺塔

```
# hanoi
const source 1
const spare 2
const dest 3
const disk_nr 0
const hanoi 32
const move 124
192 5 0 disk_nr # 5
192 0 0 source # 0
192 1 0 spare # 1
192 2 0 dest # 2
#-----------------------------
128 0 spare 9 # push spare
128 0 dest 9 # push dest
128 0 source 9 # push source
128 0 disk_nr 9 # push disk_nr
# -----------------------------
# func move
# if disk_nr==1  jmp call move
32+64 disk_nr 1 move #!!!!!!
```

```
# else
# move 1
128 0 dest 9 # push dest
128 0 spare 9 # push spare
128 0 source 9 # push source
64+1 disk_nr 1 9 # push disk_nr-1

128 0 dest 4
128 0 spare dest # update dest
128 0 4 spare # update spare
128 0 source source # update source
64+1 disk_nr 1 disk_nr # update disk_nr-1

# call func move
6 0 0 hanoi # !!!!!!!!!!!!!!!
#move 2
6 0 0 move # call move !!!!!!!!!
# move 3
128 0 source 9 # push source
128 0 dest 9 # push dest
128 0 spare 9 # push spare
64+1 disk_nr 1 9 # push disk_nr-1

128 0 source 4
128 0 spare source # update source
128 0 4 spare # update spare
128 0 dest dest # update dest
64+1 disk_nr 1 disk_nr # update disk_nr-1

# call func move
6 0 0 hanoi # !!!!!!!!!!!!!!!!!!
7 0 0 0
#----------------------
## move disk from source to dest
64 9 0 disk_nr #pop
64 9 0 source #pop
64 9 0 dest #pop
```

```
64 9 0 spare #pop
64 source 0 7 # source to 7
192 5 0 7 # lift
64 dest 0 7 # dest to 7
192 5 0 7 # lower
# ret
7 0 0 0
```